Buletin Teknik Elektro dan Informatika
(Bulletin of Electrical Engineering and Informatics)
Vol.2, No.2, June 2013, pp. 132~140
ISSN: 2089-3191

GASA-JOSH: A Hybrid Evolutionary-Annealing Approach for Job-Shop Scheduling Problem

Somayeh Kalantari*[1], Mohamad Saniee Abadeh[2]
[1]Department of Electrical, IT and Computer, Islamic Azad University, Qazvin Branch, Qazvin, Iran
[2]Department of computer, Tarbiat Modares University, Tehran, Iran
*Corresponding author, e-mail: sk_kalantari@yahoo.com[1], saniee@modares.ac.ir[2]

Abstract

The job-shop scheduling problem is well known for its complexity as an NP-hard problem. We have considered JSSPs with an objective of minimizing makespan. In this paper, we develope a hybrid approach for solving JSSPs called GASA-JOSH. In GASA-JOSH, the population is divided in non-cooperative groups. Each group must refer to a method pool and choose genetic algorithm or simulated annealing to solve the problem. The best result of each group is maintained in a solution set, and then the best solution to the whole population is chosen among the elements of the solution set and reported as outcome. The proposed approach have been compared with other algorithms for job-shop scheduling and evaluated with satisfactory results on a large set of JSSPs derived from classical job-shop scheduling benchmarks. We have solved 23 benchmark problems and compared results obtained with a number of algorithms established in the literature.

Keywords: *job-shop scheduling, genetic algorithm, simulated annealing*

1. Introduction

Scheduling is one of the most important issues in the planning and operation of manufacturing systems [1]. Bruker and Schile [2] were the first to address this problem in 1990. They developed a polynomial graphical algorithm for a two-job problem. Exact algorithms are not effective for solving JSP and large instances [3]. Several procedures such as dispatching rules, local searches and meta-heuristics (such as taboo search, simulated annealing and genetic algorithm) have been developed in recent years [4].

GA is an effective meta-heuristic to solve combinatorial optimization problems, and has been successfully adopted to solve the JSP. Recently more and more papers are talking about this topic. They differ from each other in encoding and decoding schemes, initial population method, and offspring generation strategy. Chen et al. [5] used an integrated approach to solve the FJSP. Yang [6] proposed a GA-based discrete dynamic programming approach. Zhang and Gen [7] proposed a multi stage operation-based genetic algorithm to deal with the problem from the point view of dynamic programming. Ho et al. [8] proposed architecture for learning and evolving of FJSP called Learnable Genetic Architecture (LEGA). Pezzella et al. [3] integrated different strategies for generating the initial population, and selecting the individuals for reproducing new individuals. Tay and Wibowo [9] combined the GA and a variable neighbourhood descent (VND) for solving the FJSP. In this paper, we aimed to solve JSP using a hybrid method. The method enjoys Genetic algorithm (GA) and Simulated annealing (SA). For which sub populations, groups, use one of the algorithms, GA or SA, arbitrarily to solve JSP.

The paper is organized as follows. Section 2 gives the description of the JSP. The proposed hybrid approach along with the structure of the work is explained in section 3. Section 4 reports benchmark instances, experimental results and Parameter settings of GASA-JOSH. In the last section we conclude the paper.

2. The Description of Job Shop Scheduling Problem

In the general JSSP, there are j jobs, job= {j1, j2 … j n}, and m machines, machine= {m1, m2… m n}. Each job comprises a set of tasks which must each be done on a different

Received November 15, 2012; Revised February 3, 2013; Accepted March 5, 2013

machine for different specified processing times in given job-dependent order. Table 1 shows a standard 6*6 benchmark problem (i.e. j=6 & m=6) from [10]. In this example, job 1 must go to machine 3 for 1 unit of time, then to machine 1 for 3 units of time, and so on. The standard job-shop scheduling problem makes the following constraints and assumptions [11]:

Table 1. The 6×6 Benchmark Problem

	(m, t)	(m, t)	(m, t)	(m, t)	(m, t)	(m, t)
Job1	3, 1	1, 3	2, 6	4, 7	6, 3	5, 6
Job2	2, 8	3, 5	5, 10	6, 10	1, 10	4, 4
Job3	3, 5	4, 4	6, 8	1, 9	2, 1	5, 7
Job4	2, 5	1, 5	3, 5	4, 3	5, 8	6, 9
Job5	3, 9	2, 3	5, 5	6, 4	1, 3	4, 1
Job6	2, 3	4, 3	6, 9	1, 10	5, 4	3, 1

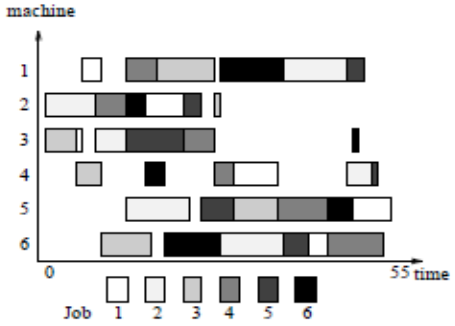


Figure 1. An Optimal Schedule for 6×6 JSSP Benchmark

- The processing time for each operation using a particular machine is defined.
- There is a pre-defined sequence of operations that has to be maintained to complete each job.
- Delivery times of the products are undefined.
- There is no setup or tardiness cost.
- A machine can process only one job at a time.
- Each job is performed on each machine only once.
- No machine can deal with more than one type of task.
- The system cannot be interrupted until each operation of each job is finished.
- No machine can halt a job and start another job before finishing the previous one.
- Each and every machine has full efficiency.

The objective considered in this work is to minimize the total elapsed time between the beginning of the first task and the completion of the last task, the makespan. Makespan is defined as Equation (1). In that equation, the completion time is denoted by C_j.

$$C_{max} = \max_{j=1\ldots n}\{C_j\} \quad (1)$$

The other measures of schedule quality exist, but shortest make span is the simplest and most widely used criterion. For the above problem the minimum make span is known to be 55 as in for example. The schedule is shown in Figure 1 [12].

3. The Proposed Hybrid Approach (GASA-JOSH)

In this approach, GASA-JOSH, using a random way the main population is created and then divided in non-cooperative groups such that group boundary is closed. The whole population incorporates individuals which are distributed in the search space globally. That is, individuals of a group are not necessarily adjacent. These groups are of equal size. A method

pool is considered which contains GA and SA method. Each group chooses a method from the pool autonomously and the process of finding the best solution in each group is done by the method selected. Evolution in the groups is carried out independently. The best result of each group is saved. The best solution to the entire population is equivalent to the best of all groups. The framework of the proposed approach, GASA-JOSH, is illustrated in Figure 2. In the following sub sections Genetic algorithm along with the simulated annealing are described.

3.1. Genetic Algorithm

GA is a local search algorithm that follows the evolution paradigm. Starting from an initial population, the algorithm applies genetic operators in order to produce off springs (in the local search terminology, it corresponds to exploring the neighbourhood) which are presumably more fit than their ancestors. At each generation, every new individual corresponds to a solution, a schedule of the given JSP instance. The strength of GA with respect to other local search algorithms is due to the fact that in a GA framework more strategies can be adopted together to find individuals to add to the mating pool, both in the initial population phase and in the dynamic generation phase. Then a more variable search space can be explored at each algorithm step [3].

Procedure GASA-JOSH (GA- SA parameters, JSP data set)→ an optimal schedule

a) *Initialize the GA parameters:* population size, number of groups (Gnum), size of each group (Gsize), maximum iteration number, crossover probability, and mutation probability.
b) *Initialize the SA* parameters*:* maximum iteration per each temperature, the first temperature, the last temperature, alpha
c) *Initialize population*, evaluate *individuals*, *divide* them by Gnum and give Gsize individuals to each group

```
While (exist group)
Group Pop= individuals of the group
Select a method from method pool(SG)
If  SG=1  then
   While (not termination condition)
                          Crossover and create Cross over Pop
                          Mutation and create Mutate Pop
                          Evaluate pop (crossover+ mutate)
                          Get new population
                          (1/2 from the best individuals of the group +
                          1/2 from the best individuals of (Mutate Pop + Cross over Pop))
                          Evaluate Group Pop
Else
       While (not termination condition)
         While (not termination per temperature condition)
                    Create neighbour of individuals as new solutions
       Evaluate new solutions
            Delta = (New Sol .Cost- Group Pop. Cost)/ Group   Pop. Cost)
       If Delta<=0 then new solution is better
       Else with the probability of  P , new solution is better
```

Figure 2. The Proposed Framework

3.1.1. Selection

The selection phase is in charge to choose the better individuals to prepare for the crossover. In this paper the roulette wheel selection is adopted.

3.1.2. Crossover

Crossover can be regarded as the backbone of genetic search. It intends to inherit nearly half of the information of two parent solutions to one or more offspring solutions.

Provided that the parents keep different aspects of high quality solutions, crossover induces a good chance to find still better offspring [13].

3.1.3. Mutation

Mutation usually works on a single chromosome and creates another chromosome through alternation of the value of a string position or exchange of the values of two string positions in order to maintain the diversity of population. In this paper, an exchange order mutation operator is used. That is, two operations are chosen randomly and then their positions are exchanged [14].

3.1.4. Stopping Criteria

Fixed number of generations is considered as stopping criteria. If the stop criterion is satisfied, the best chromosome is given as output and the algorithm ends.

3.2. Simulated Annealing

Simulated annealing (SA) is motivated by an analogy to annealing in solids. The idea of SA comes from a paper published by Metropolis et al. in 1953 [15]. The algorithm in that paper simulated the cooling of material in a heat bath. In 1982, Kirkpatrick et al. [16] took the idea of the Metropolis's algorithm and applied it to optimization problems. The idea is to use simulated annealing to search for feasible solutions and converge to an optimal solution [17].

The objective function is considered as a measure of the energy of the system and this is maintained fixed for a certain number of iterations (a temperature cycle). In each of the iterations, the parameters are changed to a nearby location in parameter space and the new objective function value calculated. If it decreased, then the new state is accepted. If it increased, then the new state is accepted with a probability that follows a Boltzmann distribution (higher temperature means higher probability of accepting the new state). After a fixed number of iterations, the stopping criterion is checked. If it does not come time to stop, then the system's temperature is reduced and the algorithm continues. Simulated annealing is a stochastic algorithm that is guaranteed to converge, if ran for an infinite number of iterations. It is one of the most robust global optimization algorithms, although it is also one of the slowest.

Figure 3 shows the simulated annealing algorithm [18]. In this Figure, the VALUE function corresponds to the total energy of the atoms in the material, and T corresponds to the temperature. The schedule determines the rate at which the temperature is lowered. Individual moves in the state space correspond to random fluctuations due to thermal noise. One can prove that if the temperature is lowered sufficiently slowly, the material will attain a lowest-energy (perfectly ordered) configuration. This corresponds to the statement that if schedule lowers *T* slowly enough, the algorithm will find a global optimum.

```
Function SIMULATED-ANNEALING (Problem, Schedule)   returns a solution state
Inputs:  Problem, a problem, Schedule, a mapping from time to temperature
Local Variables:  Current, anode
                  Next, a node
                  T, a “temperature” controlling the probability of downward steps
Current= MAKE-NODE (INITIAL-STATE[Problem])
For   t=1 to ∞ do
    T= Schedule(t)
    If T=0 then return Current
    Next= a randomly selected successor of Current
    ΔE=VALUE[Next]-VALUE[Current]
    If ΔE>0 then Current=Next
    Else Current=Next only with probability exp(-ΔE/T)
```

Figure 3. Simulated Annealing Algorithm

4. Computational Experiments

4.1. Benchmark Instances

The benchmark instances considered in the experiments are summarized in Table 2. In that table, the first column shows the instance name and the second one shows the relevant literature specification.

Table 2. Benchmark Instances

Instance	Literature
ft06-ft10- ft20	Muth, J. F. & Thompson, G. L. (1963)"Industrial Scheduling", Prentice Hall, Englewood Cliffs, New Jersey, pp225-251.
la01-la40	Lawrence, S. (1984)"Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (Supplement)", Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania.
abz5-abz9	Adams, J. & Balas, E. & Zawack, D. (1988)"The shifting bottleneck procedure for job shop scheduling", Management Science 34, pp391-401.
orb01-orb10	Applegate, D. & Cook, W. (1991)" A computational study of the job-shop scheduling instance", ORSA Journal on Computing 3, pp149-156. (they were generated in Bonn in 1986)
swv01-swv20	Storer, R.H. & Wu, S.D. & Vaccari R. (1992)"New search spaces for sequencing instances with application to job shop scheduling", Management Science 38, pp1495-1509.
yn1-yn4	Yamada, T. & Nakano, R. (1992)"A genetic algorithm applicable to large-scale job-shop instances", North-Holland, Amsterdam, pp281-290.

4.2. Features of the Computing Device Used and Parameters Setting

GASA-JOSH was implemented in Matlab. The tests were run on a computer with a 2.8GHz Pentium five CPU and 2GB RAM on Microsoft windows XP operating system. The parameters used in this approach are chosen experimentally to get a satisfactory solution in an acceptable time span. Through experimentation, name and value of the parameters considered are shown in Table 3.

Table 3. Parameter Settings

Parameter Name	value
Number of generations	150
Maximum number of iterations per temperature	10
Number of groups	6
Size of each group	100
Size of the Population	600
Probability of Crossover	0.8
Probability of Mutation	0.01
Initial Temperature	10
Final Temperature	0.001
Cooling Factor	0.91

4.3. Experimental results

In this paper, in order to give a rough idea about the quality achieved, we confine to the 23 famous problems of Muth & Thompson [10] and Lawrence [19]. We ran the proposed approach five times on the same instance to obtain meaningful results. The results appear in Table 4. It respectively lists problem name, problem size (number of jobs × number of operations), the best known solution (BKS), and the best solution obtained in this paper (GASA-JOSH). The solutions obtained by the following literature: Park et al. [20], Gao et al. [21], and Yang et al. [22] along with the relative deviations, Dev (%), are shown in the next columns of the table. Dev (%) columns imply the relative deviation of those algorithms with respect to GASA-JOSH. The relative deviation is defined in Equation (2).

$$Dev = \left[\frac{Mk_{comp} - Mk_{GASA-JOSH}}{Mk_{comp}}\right] \times 100 \quad (2)$$

Where, $MK_{GASA\text{-}JOSH}$ is the makespan obtained by our approach, and MK_{comp} is the makespan of the algorithm we compared to. Results show that for 78.26% of the cases, the underlined numbers shown in GASA-JOSH column of Table 3, our approach outperforms the other algorithms.

GASA-JOSH is firstly tested on Lawrence's data set (LA01-LA20). Applied instances of this data set consist of 20 problems with 10, 15 or 20 jobs, 5 or 10 machines, and 5 or 10 operations. Secondly, it is tested on Thompson's data set (Ft06-Ft10-Ft20). The number of jobs in that data set ranges from 6 to 20, the number of machines ranges from 5 to 10, and the number of operations for each job ranges from 5 to 10.

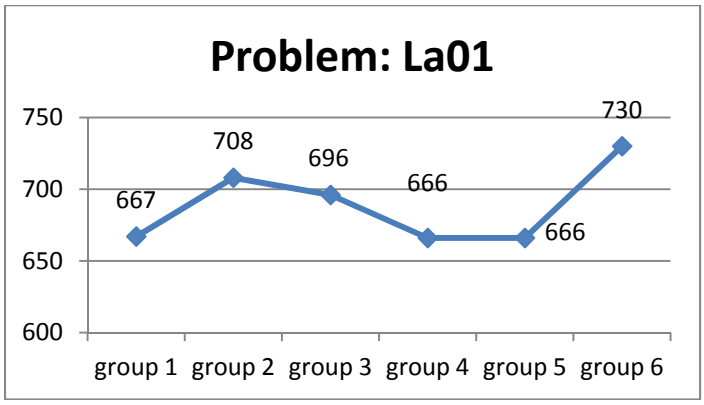


Figure 4. Makespan of the Proposed Approach Obtained in Each Group for La01

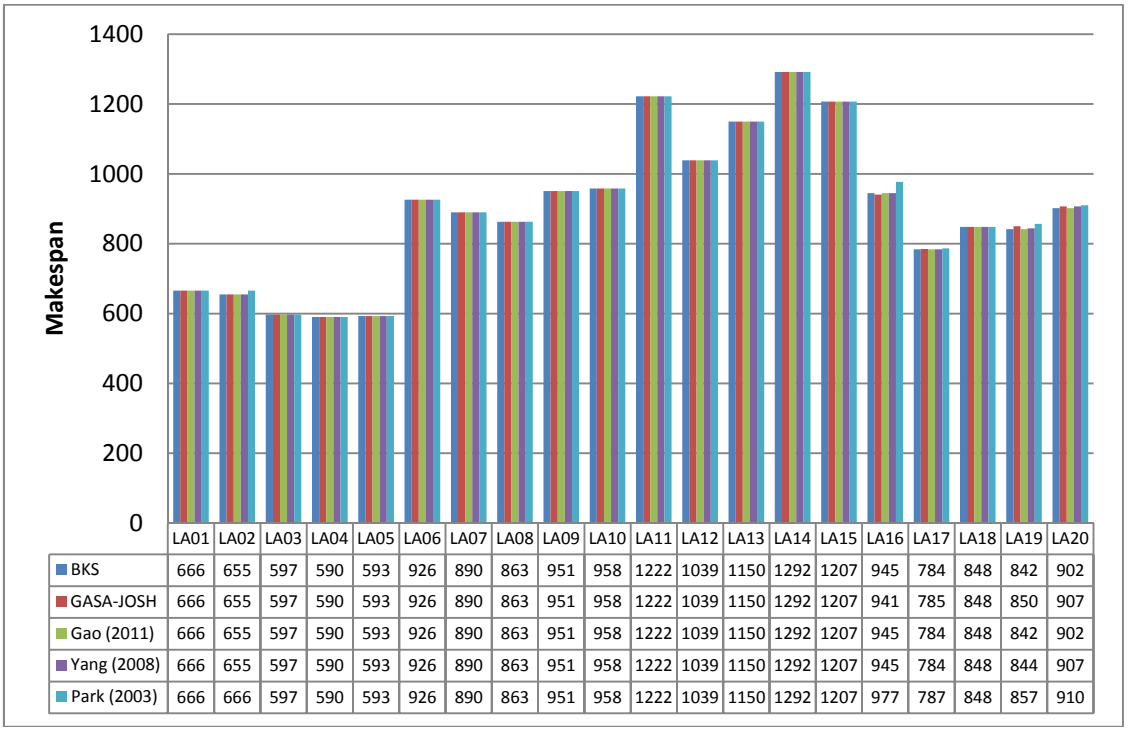
Table 4. Experimental Results

Problem	Size	BKS	GASA-JOSH	Gao (2011)	Dev (%)	Yang (2008)	Dev (%)	Park (2003)	Dev (%)
FT06	6×6	55	55	55	0	55	0	55	0
FT10	10×10	930	943	930	-1.4	930	-1.4	936	-0.75
FT20	20×5	1165	1178	1165	-1.12	1165	-1.12	1177	-0.08
LA01	10×5	666	666	666	0	666	0	666	0
LA02	10×5	655	655	655	0	655	0	666	1.65
LA03	10×5	597	597	597	0	597	0	597	0
LA04	10×5	590	590	590	0	590	0	590	0
LA05	10×5	593	593	593	0	593	0	593	0
LA06	15×5	926	926	926	0	926	0	926	0
LA07	15×5	890	890	890	0	890	0	890	0
LA08	15×5	863	863	863	0	863	0	863	0
LA09	15×5	951	951	951	0	951	0	951	0
LA10	15×5	958	958	958	0	958	0	958	0
LA11	20×5	1222	1222	1222	0	1222	0	1222	0
LA12	20×5	1039	1039	1039	0	1039	0	1039	0
LA13	20×5	1150	1150	1150	0	1150	0	1150	0
LA14	20×5	1292	1292	1292	0	1292	0	1292	0
LA15	20×5	1207	1207	1207	0	1207	0	1207	0
LA16	10×10	945	941	945	0.42	945	0.42	977	3.68
LA17	10×10	784	785	784	-0.13	784	-0.13	787	0.25
LA18	10×10	848	848	848	0	848	0	848	0
LA19	10×10	842	850	842	-0.95	844	-0.71	857	0.82
LA20	10×10	902	907	902	-0.55	907	0	910	0.33
Average improvement					+4.21		-1.21		-0.42

In Table 4, the compared computational results show that for LA16 the best makespan of GASA-JOSH over five runs is better than that of the others. For LA01-LA15, and LA18 the proposed approach could gain the same good results as the other literature. In Figure 4, we draw the best makespan value obtained by each group of the whole population over five runs for the La01 problem with 10 jobs and 5 machines. For this problem the best known makespan in the other literature is 666. As said before, the second test of the approach was run on Muth &

Thompson instances (Ft06, Ft10, and Ft20). Ft06 was solved to optimality (best makespan =55). Meanwhile, Ft10 and Ft20 were respectively solved with makespan of 943 and 1178. Makespan of the best known solution, makespan of the best GASA-JOSH solution, and a comparison with the other three approaches addressed in Table 4 are shown in Figures 5 and 6 respectively for Lawrence and Thompson datasets. Figure 7 shows the relative deviation of the best known makespan with respect to GASA-JOSH.

To summarize, for 17 out of 23 problems GASA-JOSH could gain the same good results as the literature and for 1 problem could gain better results than the other papers.



	LA01	LA02	LA03	LA04	LA05	LA06	LA07	LA08	LA09	LA10	LA11	LA12	LA13	LA14	LA15	LA16	LA17	LA18	LA19	LA20
BKS	666	655	597	590	593	926	890	863	951	958	1222	1039	1150	1292	1207	945	784	848	842	902
GASA-JOSH	666	655	597	590	593	926	890	863	951	958	1222	1039	1150	1292	1207	941	785	848	850	907
Gao (2011)	666	655	597	590	593	926	890	863	951	958	1222	1039	1150	1292	1207	945	784	848	842	902
Yang (2008)	666	655	597	590	593	926	890	863	951	958	1222	1039	1150	1292	1207	945	784	848	844	907
Park (2003)	666	666	597	590	593	926	890	863	951	958	1222	1039	1150	1292	1207	977	787	848	857	910

Figure 5. Makespan Comparison Chart for La01-La20

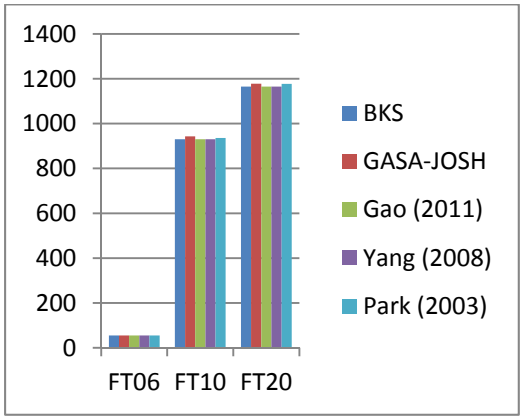


Figure 6. Makespan Comparison Chart for Ft06, Ft10, and Ft20

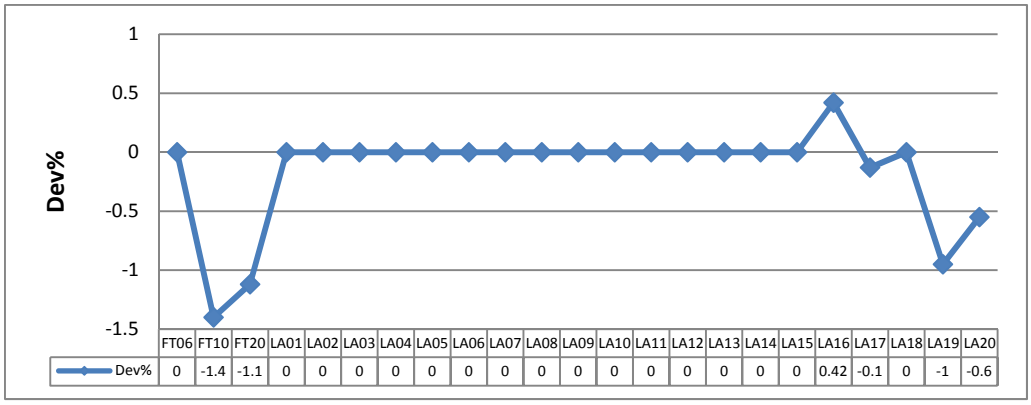


Figure 7. The Relative Deviation of the Best Known Makespan with Respect to GASA-JOSH

5. Conclusion

In this paper, a new evolutionary-annealing approach (GASA-JOSH) uses Genetic algorithm and simulated annealing algorithm for solving job-shop scheduling problem is proposed. The approach is compared with some algorithms in the literature. It is tested on a set of 23 standard instances and compared with 3 other approaches. The computational results show that the approach, GASA-JOSH, could produce the Best Known Solution on 78.26% of all instances tested. That is it could gain results which are better or equal to the BKS on 18 out of 23 problems.

References

[1] Zhang G, Gao L, Shi Y. A *genetic algorithm and taboo search for solving flexible job shop schedules. P*roc. computational intelligence and design International symposium. 2008; 369-372.
[2] Brucker P, Schile R. Job-shop scheduling with multi-purpose machines. *Computing*. 1990; 45(4): 369-375.
[3] Pezzella F, Morganti G, Ciaschetti G. A genetic algorithm for the flexible job-shop scheduling problem. *Computers and Operations Research*. 2007; 5(10): 3202–3212.
[4] Zhang G, Gao L, Shi Y. An effective genetic algorithm for the flexible job-shop scheduling problem. *Expert system with applications*. 2011; 38: 3563-3573.
[5] Chen H, Ihlow J, Lehmann C. A genetic algorithm for flexible job shop scheduling. *IEEE international robotics and automation conference*. 1999; 1120-1125.
[6] Yang JB. GA-based discrete dynamic programming approach for scheduling in FMS environments. *IEEE Transaction on Systems, Man, and Cybernetics, Part B*. 2001; 31(5): 824–835.
[7] Zhang HP, Gen M. Multistage-based genetic algorithm for flexible job shop scheduling problem", Journal of Complexity International. 2005; 48: 409–425.
[8] Ho NB. Tay JC, Edmund M, Lai K. an effective architecture for learning and evolving flexible job shop schedules. *European Journal of Operational Research*. 2007; 179: 316–333.
[9] Tay JC, Wibowo D. An effective chromosome representation for evolving flexible job shop schedules", GECCO 2004, Lecture notes in computer Science. 2004; 3103: 210–221.
[10] Muth JF, Thompson G. Industrial Scheduling, Prentice Hall, Englewood Cliffs, New Jersey. 1963; 225-251.
[11] Giovanni L, Pezzella F. An improved genetic algorithm for the distributed and flexible job shop scheduling problem. *European journal of operational research*. 2010; 200: 395-408.
[12] Fang H, Ross P, Corne D. *A promising genetic algorithm approach to: job shop scheduling. rescheduling and open shop scheduling problems.* Fifth international conference on genetic algorithms. 1993; 375-382.
[13] Bierwirth C. A generalized permutation approach to job shop scheduling with genetic algorithms. *OR Spectrum*. 1995; 1787-1792.
[14] Xing YJ, Wang ZQ, Sun J, Meng J. A multi objective fuzzy genetic algorithm for job shop scheduling problems. *Journal of achievements in materials and manufacturing engineering*. 2006; 17: 297-300.
[15] Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller A, Teller E. Equation of State Calculation by Fast Computing Machines. *J. of Chem. Phys*. 1953; 21: 1087-1091.
[16] Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by Simulated Annealing. *Journal of Science*. 1983; 220: 671-680.

[17] Kendall G. http://www.cs.nott.ac.uk/ ~gxk/aim/notes/simulatedannealing.doc.
[18] Russell S, Norvig P. Artificial Intelligence: A Modern Approach, Prentice Hall. 1995.
[19] Lawrence S. Supplement to resource constrained project scheduling: An experimental investigation of heuristic scheduling techniques. Pittsburgh, PA: GSIA, Carnegie Mellon University. 1984.
[20] Park BJ, Choi HR, Kim HS. A hybrid genetic algorithm for the job shop scheduling problems. *J of Computers & Industrial Engineering*. 2003; 45(4): 597–613.
[21] Gao L, Zhang G, Zhang L, Li X. An effective memetic algorithm for solving the job- shop scheduling problem, journal of Computers and Industrial engineering. 2011; 60: 699-705.
[22] Yang J, Sun L, Lee H, Qian Y, Liang Y. Clonal Selection Based Memetic Algorithm for Job Shop Scheduling Problems. *Journal of Bionic Engineering. 2008; 5: 111-119.*